Goal Modeling Education with GRL: Experience Report

Daniel Amyot

School of Electrical Engineering and Computer Science
University of Ottawa, Ottawa, Canada
damyot@uottawa.ca

Abstract. Goal modeling and analysis with the Goal-oriented Requirement Language (GRL) is taught in software engineering and computer science at the University of Ottawa since 2003. This paper presents the general education approach taken in an undergraduate requirements engineering course and in a graduate software engineering course. Some of the particularities of these courses involve the use of a general GRL modeling pattern, the combined use with Use Case Maps (for operationalization and for business processes), the coverage of qualitative and quantitative analysis approaches, the use of indicators, and automated evaluations of strategies supported by the jUCMNav tool. This paper also reflects on some successes and difficulties observed in the past decade while teaching these concepts.

Keywords: Education · Experience · Goal-oriented Requirement Language · jUCMNav

1 Introduction

Goal modeling is taught at the University of Ottawa since 2003. The Goal-oriented Requirement Language (GRL), part of the User Requirements Notation (URN) [1,7], is the language being taught in the following courses, with a total audience of well over 1,000 students:

- *Introduction to Software Engineering* (2003-2004), undergraduate, computer science program, 3rd-year, without tool support or labs.
- *Software Requirements Analysis* (2005-2014), undergraduate, 3rd-year, software engineering program, with tool support and labs.
- *Software Engineering* (10 times between 2004 and 2015), graduate, masters and Ph.D., computer science program, with tool support but no labs.

In the above semester-long courses, goal modeling and analysis with GRL is the topic of a 3-hour lecture. In addition, one course has a 3-hour laboratory where students can learn jUCMNav, an Eclipse plug-in for GRL modeling and analysis, with the help of a teaching assistant [6]. The same laboratory material (tutorial on the construction of a

goal model and strategies, with an additional exercise) is made available to the students of the graduate course, but for self-study only. All courses get an assignment with a GRL modeling and analysis problem (textual description) that requires the use of jUCMNav. Another (shorter) problem is always present in a partial/final exam.

This paper reports on my experience teaching GRL, mainly for the past decade (the last two courses) because of the availability of tools. The focus here is on the styles of models being taught (section 2) and on the types of analyses they enable (section 3), together with the informal feedback I perceived from students during the courses.

2 Teaching and Learning GRL Modeling

GRL is a rich goal-oriented modeling language that can be used for different purposes, from social modeling to decision making and rationale documentation. The courses introduce GRL with the later part. After a bird's eye introduction to URN, students are taught the need for documenting design rationales in software. *Rationales* are often described with comments in the code and in repository commits in practice. They are also often represented in a tabular way, where different options (rows) are compared against different criteria (columns), using a coarse-grained qualitative scale or a quantitative scale with some aggregation function (e.g., weighted sum). Students often see such tables in magazines and web sites. They then learn that such tabular representation only captures a partial view of reality: dependencies between criteria are not shown, and "who is concerned with what criteria" is not described either. With these limitations in mind, the concept of goal and its use in software engineering are introduced, followed by the syntax of GRL based on a security example.

This quickly leads to the introduction of a *GRL pattern* (Fig. 1) that describes general decision making for systems with alternatives.

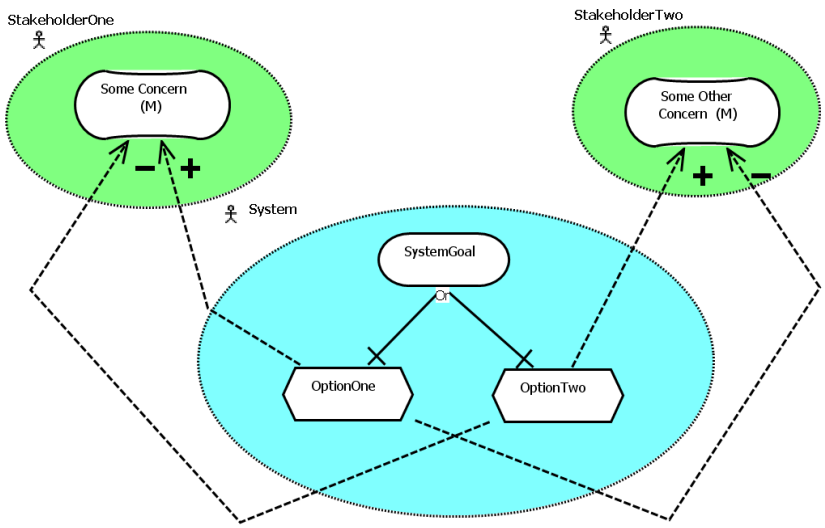

Fig. 1. Common GRL pattern for decision making in context

As shown in this figure, system functionalities are captured with *goals* (that can be AND-decomposed). The various means of achieving these goals are captured as *tasks* linked with OR-decompositions. *Actors* capture system stakeholders and *softgoals* represent their concerns, often non-functional in nature. Some of the alternative tasks will also have some positive or negative impacts on some of the stakeholder concerns.

With such a pattern, the students learn that global decisions (one for each OR-decomposition, in order to satisfy system goals) become non trivial and that it is difficult to satisfy all actors (leading to the notion of *trade-off*). They also learn that once agreed on, a global decision (defined as a *strategy* in GRL) combined with the model itself document the "why" aspect of the system, which is a view absent from UML.

GRL supports quantitative scales for contributions ([-100..100]), for satisfaction values, and for the importance of intentional elements to their containing actor. There are also qualitative scales for contributions {Break, ..., Make}, satisfactions {Denied, …, Satisfied}, and importance {None, …, High}. Students generally understand that a qualitative scale is used in the early modeling steps, when little information is available. As the understanding of the problem and of the potential impact of solutions improves, the modeler can move to a more fine-grained quantitative scale.

However, in a quantitative context, it becomes difficult to find proper values (e.g., should the weight of a contribution be 30 or 40?). Although jUCMNav and GRL contain some features that can help cope with such decisions, e.g., value ranges and contribution overrides [3], there is neither time to cover these advanced features nor their other usages (e.g., for sensitivity analysis). Yet, some time is devoted to the coverage of GRL *indicators* as a means to better root part of the goal model in reality, and also to connect GRL to business process modeling (where managers use indicators to monitor systems and satisfaction). In GRL, an indicator converts an observable value (in a real unit like kilometers or Euros) to a GRL satisfaction value through a comparison with target, threshold, and worst-case values. Students learn to convert uncertain contribution weights into more meaningful indicators with a contribution of weight 100. This helps remove some of the uncertainty and make models more falsifiable [8].

URN combines GRL with *Use Case Maps* (UCM), a notation for causal scenarios superimposed onto a component structure [7]. In both courses, a second 3-hour lecture is devoted to UCM (with jUCMNav support). Students learn to use UCM to operationalize some of the tasks and goals found in GRL models. They learn that both views are needed for requirements engineering activities and for business process modeling as the "why" and "how much" aspects are uniquely covered by GRL and the "when" aspects are uniquely covered by UCM (both cover "what", "who" and "where" aspects). They also learn that both views can be developed in parallel and iteratively as some stakeholders (e.g., managers) tend to discuss more in terms of objectives (at the GRL level) whereas others involved in operations will describe their knowledge and needs more in terms of UCM-like concepts. Finally, they also realize that creating traceability links between elements of the GRL and UCM views can help answer *consistency* and *completeness* questions (e.g., why keep a scenario without a goal/purpose? Why is this functional goal not refined by any scenario?). All of these concepts are illustrated with intuitive examples from the telecommunication domain, discussed briefly in [1].

The complexity of goal models can quickly become an issue [5], and this is also the case in GRL (as it is in *i**) because the language lacks modularity constructs. In jUCMNav, complexity is managed through having different diagrams (views) as part of one model, where the elements and links can be referenced in multiple views. jUCMNav supports many ways of navigating between references of an intentional element or actor. Students learn to decompose complex models with many diagrams.

Unlike *i**, GRL does not distinguish explicitly between *strategic dependency* (SD) diagrams and *strategic rationale* (SR) diagrams. These views can be distinct in GRL, but they are often intertwined. While several other courses often start with SD diagrams to introduce goal modeling [3], our courses focus first on the SR view, and covers GRL dependency links later, in a brief introduction to social modeling. One lesson learned from experience is that computer science and software engineering students see the value of SR views right away, but they do not see the value of SD views in the limited time available to cover goal modeling.

Although the students can distinguish between goals, softgoals and tasks fairly well, means-end links are no longer taught (as OR-decomposition is used instead). Correlation links are mentioned but not taught (contributions are used instead), and resources and beliefs are barely mentioned. In the early years, all of these concepts were covered, but the students were puzzled by the need for so many types of intentional elements and links. As students were using them mostly incorrectly, their coverage was simply minimized, without a real negative impact on the expressiveness of resulting models. This subset shares many commonalities with what is being taught by Dalpiaz in his first-year course for information system students [6].

On a couple of years, both at the undergraduate and graduate levels, my colleagues and I attempted to discuss other goal modeling languages (especially *i** and KAOS) and compare them with GRL, but this did not raise any real interest. I suspect this is because small differences between little-known languages do not become attractive until one goal language is actually mastered. Now we only mention their existence.

3 Teaching and Learning GRL Analysis

The teaching of analysis techniques is done iteratively and is interleaved with modeling. Once the GRL syntax is introduced, jUCMNav is used to create a model on the fly based on suggestions of the students on a domain they know (e.g., a university registration system). Once a few actors, their intentions, and some links (especially OR-decompositions) are available, the notion of "what-if" analysis is introduced. In GRL, what-if situations are captured with *strategies*, which are initial satisfaction values assigned to some of the intentional elements. A strategy is evaluated via a propagation algorithm, and several qualitative and quantitative ones are supported in jUCMNav [2]. The semantics of the various GRL links is revisited, this time in a more formal way based on the qualitative and quantitative propagation of satisfaction values (including factor evaluations). One lesson learned here is that a better understanding of how these algorithms work leads to a better and more consistent selection of GRL relationships (e.g., decomposition versus contributions) by students.

A related lesson learned is that students in these courses enjoy *automated* analysis and have little interest in manual or interactive propagation, maybe because of the immediate feedback and low effort coming with automation. With jUCMNav, one can easily describe many strategies, and the analyst can go from one to the next with instantaneous evaluation feedback. This is not something they could do with manual or interactive propagation. Conflicts can still be detected (e.g., via rules in the Object Constraint Language, automatically checked by jUCMNav) and then resolved in the strategy by explicitly setting the (resolved) satisfaction value of the intentional element under conflict. Students also learn to compare strategy results in different ways:

- By alternating between the graphical evaluations of the strategies of interest;
- By using a *strategy diff* feature in jUCMNav, which shows the deltas in the evaluation of a strategy on the model compared to that of a base strategy [3];
- By exporting the resulting evaluations of strategies as an Excel/CSV file or by generating a tabular report in HTML or Word using jUCMNav.

Special attention is dedicated to *trade-off analysis* with the help of pre-existing examples. Students learn how to create a reasonable set of potential strategies and select the "best one" based on which actor(s) we want to prioritize in terms of satisfaction. One challenge here is to come up with a set of strategies that is reasonable while not being exhaustive (as sometimes the number of potential combinations explodes rapidly). This is a problem akin to test creation and selection in software testing.

In a different lecture of the graduate course, students are introduced to *aspect-oriented modeling* (AOM), and GRL and UCM are revisited in that context. Students are taught the benefits of aspects for handling cross-cutting concerns, with examples exploiting an aspect-oriented extension of URN [9]. Aspect-oriented GRL is briefly introduced, but not tested in assignments or exams (only aspect-oriented UCM is used in assignments and exams, because of the existence of partial tool support in jUCMNav). The value of aspect-oriented modeling and analysis is also clearer in a UCM context than in a GRL context, because cross-cutting concerns are less frequent at the goal level than in lower-level operational details.

The graduate students are also given another lecture on the *Object Constraint Language* (OCL), and again this is an opportunity to revisit GRL. jUCMNav has over one hundred predefined static semantic rules described with OCL, and it offers a user interface to select which ones to apply to a model, and even to create new ones [4]. There are, for example, rules use to restrict the use of the GRL language to a modeling style compatible with *i**. OCL is also used to compute various metrics on GRL models. Students are briefly taught about the URN metamodel, and how to create new rules or metrics using OCL and jUCMNav. This is further tested in an assignment.

4 Conclusions and Future Work

This paper provided a brief overview of the GRL education material taught in courses at the University of Ottawa, together with my perception of the student feedback on their learning experience related to modeling and analysis of goal models with GRL.

Students of the undergraduate course have to do a semester-long project in teams, leading to the creation and validation of a software requirements specification for real

stakeholders. One encouraging result is that some teams voluntarily choose to use GRL to model stakeholders' objectives together with the intended functionality of the system, and with alternatives defined and selected through "what-if" analysis of strategies. Similarly in the graduate course, where teams of students need to compare two software engineering tools in a given business context, several teams use GRL to capture stakeholder objectives and the impact of choosing one tool over the other one.

There are yet many opportunities to improve the learning experience of students in these courses. For one, there is a major lack of online video material (e.g., on YouTube) teaching goal modeling with GRL (or other languages), including tool support. We have a series of YouTube videos that were produced in 2010 to explain UCM modeling with jUCMNav. Although they totalize only about half an hour, these videos are much appreciated by the students, who wonder why there is nothing equivalent on the GRL side. Students also ask for more exercises that are smaller than those found in the assignment, with solutions, to better measure their understanding.

Future work is also needed on understanding the best subset of GRL to first introduce, and whether this would be different for undergraduate and graduate students.

Acknowledgments. The undergraduate courses discussed here have been taught in collaboration with at least four other professors over the years (G. v. Bochmann, O. Kabranov, S. Somé, and G. Mussbacher), partly because of the existence of multiple French/English sections and because of sabbatical years. I would like to thank them for a fruitful collaboration over the years. I also thank our numerous teaching assistants who have contributed to the education of GRL and jUCMNav.

References

1. Amyot, D., Mussbacher, G.: User Requirements Notation: The First Ten Years, The Next Ten Years. Journal of Software (JSW), Vol. 6, No. 5, 747–768 (2011)
2. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating Goal Models within the Goal-oriented Requirement Language. International Journal of Intelligent Systems (IJIS), Vol. 25, Issue 8, 841–877 (2010)
3. Amyot, D., et al.: Towards Advanced Goal Model Analysis with jUCMNav. ER Workshops 2012, LNCS 7518, Springer, 201–210. http://softwareengineering.ca/jucmnav (2012)
4. Amyot, D., Yan, J.B.: Flexible Verification of User-Defined Semantic Constraints in Modelling Tools. CASCON 2008. ACM Press, 81–95 (2008)
5. Babar, Z., Nalchigar, S., Lessard, L., Horkoff, J., Yu, E.: Instructional Experiences with Modeling and Analysis using the *i** Framework. 1st Int. iStar Teaching Workshop. CEUR-WS, Vol. 1370, 31–36 (2015)
6. Dalpiaz, F.: Teaching Goal Modeling in Undergraduate Education. 1st Int. iStar Teaching Workshop. CEUR-WS, Vol. 1370, 1–6 (2015)
7. ITU-T: Recommendation Z.151 (10/12): User Requirements Notation (URN) – Language definition. Geneva, Switzerland (2012)
8. Letier, E., Stefan, D., Barr, E.T.: Uncertainty, risk, and information value in software requirements and architecture. ICSE 2014. IEEE CS, 883–894 (2014)
9. Mussbacher, G., Amyot, D. Araújo, J., Moreira, A.: Requirements Modeling with the Aspect-oriented User Requirements Notation (AoURN): A Case Study. Transactions on Aspect-Oriented Software Development VII, LNCS 6210, Springer, 23–68 (2010)